NutriFlow

Sam Graler & Randy Hucker

Presentation Overview

- Team Info
- Goals
  - Background
  - Goals/User Stories
- Intellectual Merits
  - Novelty
- Broader Impacts
- Design Specifications
  - Design Diagrams
  - High-level Implementation Details
- Technologies
- Milestones
- Results
- Challenges

Team Information

Team Members / Division of Work

Sam Graler

Product Manager/Full-Stack Developer
Email: gralersm@mail.uc.edu

- ➔ Cart & Pantry management algorithms
- ➔ API design
- ➔ Meal/intake views
- ➔ AI touchpoints
- ➔ QA testing

Randy Hucker

Lead Develop/Backend Architect
Email: randalhucker@gmail.com

- ➔ Backend data layer
- ➔ Database models
- ➔ Store integrations
- ➔ OCR ingestion
- ➔ AI touchpoints

Dr. William Hawkins

Faculty Advisor
hawkinwh@ucmail.uc.edu

- ➔ Advisement as needed

Task/Milestone ownership was divided equally between the engineers, although most tasks are handled jointly in some capacity

Goals

Project Background

"The original idea behind this project came from frustration that Randy and I experienced while iterating on a shared spreadsheet that we were trying to use as a flexible, robust meal planner/grocery list manager/macro calculator to better pursue our fitness-related nutrition goals. After spending time trying to bend that spreadsheet to our will, we realized we could better realize our vision by creating an entirely new platform to be everything we wished our beloved spreadsheet could be. Hence, NutriFlow, the unified nutrition management platform was born" - Sam, NutriFlow cofounder

Purpose

NutriFlow is a unified nutrition and grocery planning pipeline designed to help households get organized, make healthier food choices, save money, and reduce waste.”

Get Organized

Create meal plans using an intuitive UI that prioritizes customization

Healthier Choices

Receive recommendations based on personal goals and nutrition targets

Save Money

Use Cart Management to track spending and discover more cost-effective options

Reduce Waste

Use Pantry Management to track food expiration/stock and avoid buying what you can’t use

Goal Use Cases (User Stories)

As a household cook

I want to build a weekly meal plan to hit nutrition targets easily.

As a grocery shopper

I want NutriFlow to auto-create my cart and compare prices.

As a family member

I want to manage our shared pantry to avoid waste.

Goal Use Cases (User Stories)

As a fitness hobbyist

I want to log meals and see daily macro roll-ups.

As a user with allergies

I want to filter ingredients by allergens and preferences.

As a foodie

I want to explore new recipes matching my diet.

Intellectual Merits

Intellectual Merits

Innovative Technical Design

- NutriFlow integrates **meal planning, nutrition tracking, pantry management, and cart generation** into a single unified system
  - This eliminates fragmentation across multiple apps or solutions.
- We also emphasize **data consistency across interconnected domains**
  - This requires careful schema design and state synchronization across our pipeline (foods → meals → meal plans → carts)

Full-Stack Engineering Contributions

- We employ a modular frontend architecture composed of reusable components, allowing us to seamlessly deploy across multiple platforms with minimal refactoring
- Carefully structuring API interactions and implementing client-side caching supports responsive, scalable user experiences without overwhelming our backend/database

Intellectual Merits

Systems Thinking

- We treat nutrition planning as a **constraint-optimization problem**, balancing:
  - User preferences
  - Budget considerations
  - Calories and macronutrients
  - Pantry and store availability
- This lays the groundwork for future algorithmic extensions (e.g., macro-aware auto-planning and meal recommendations).

Novel Project Scope

- Goes beyond typical calorie trackers by modeling the **entire food lifecycle**:
- Demonstrates applied knowledge of:
  - Frontend architecture
  - API design
  - State management
  - Data modeling
  - UX for complex workflows

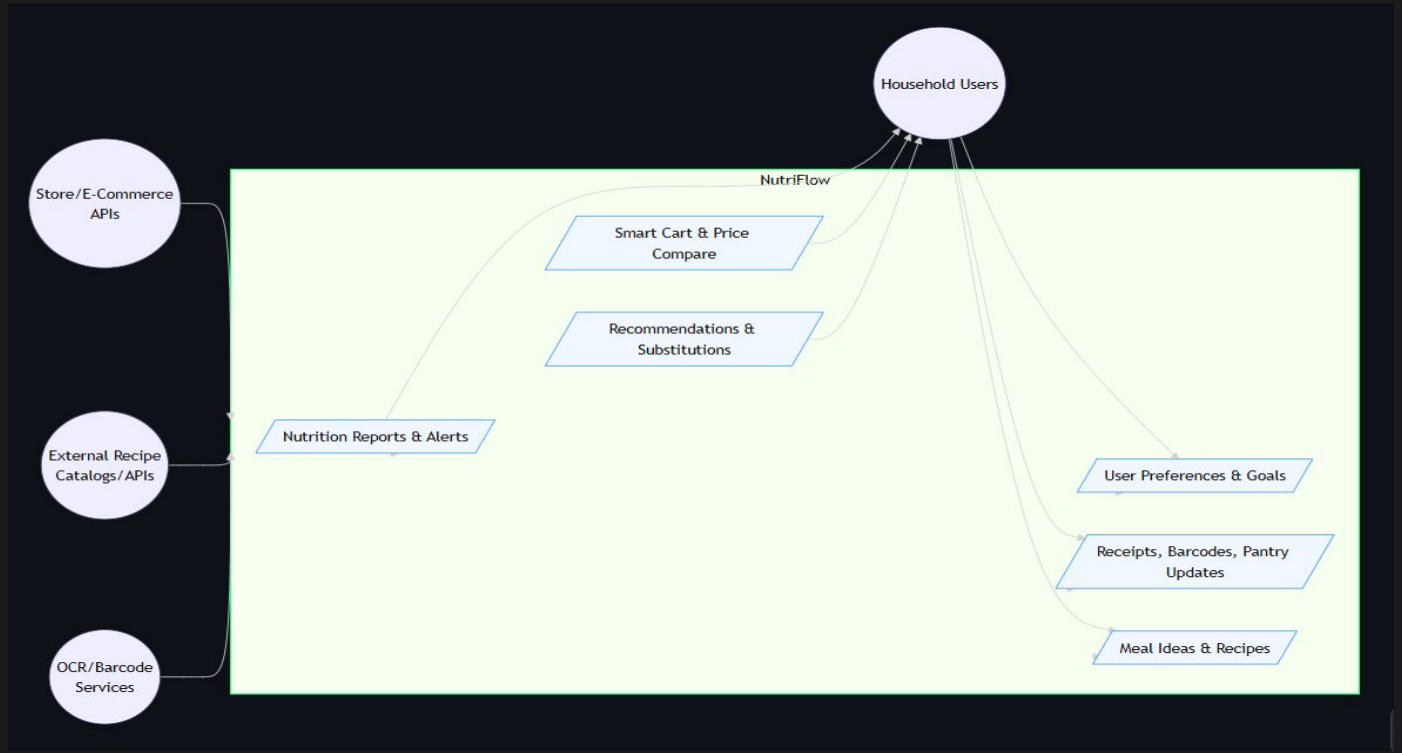
Broader Impacts

Broader Impacts

- **Supporting Healthier Lifestyles**
  - Helps users make **data-driven nutrition decisions**, encouraging consistent meal planning, macro awareness, and reduced reliance on impulsive food choices
  - Lowers the barrier to structured eating for students, busy professionals, and nutrition newcomers
- **Reducing Food Waste/Cost**
  - Pantry-aware cart generation promotes buying only what is needed, using existing ingredients, and more efficient grocery shopping
  - Supports budget-conscious planning and minimizes unnecessary purchases
- **Accessibility and Practical Impact**
  - Designed to be usable in a casual manner while still offering advanced capabilities for users with high-performance goals
  - Centralizes tools that are typically scattered across multiple platforms, improving accessibility to nutrition planning

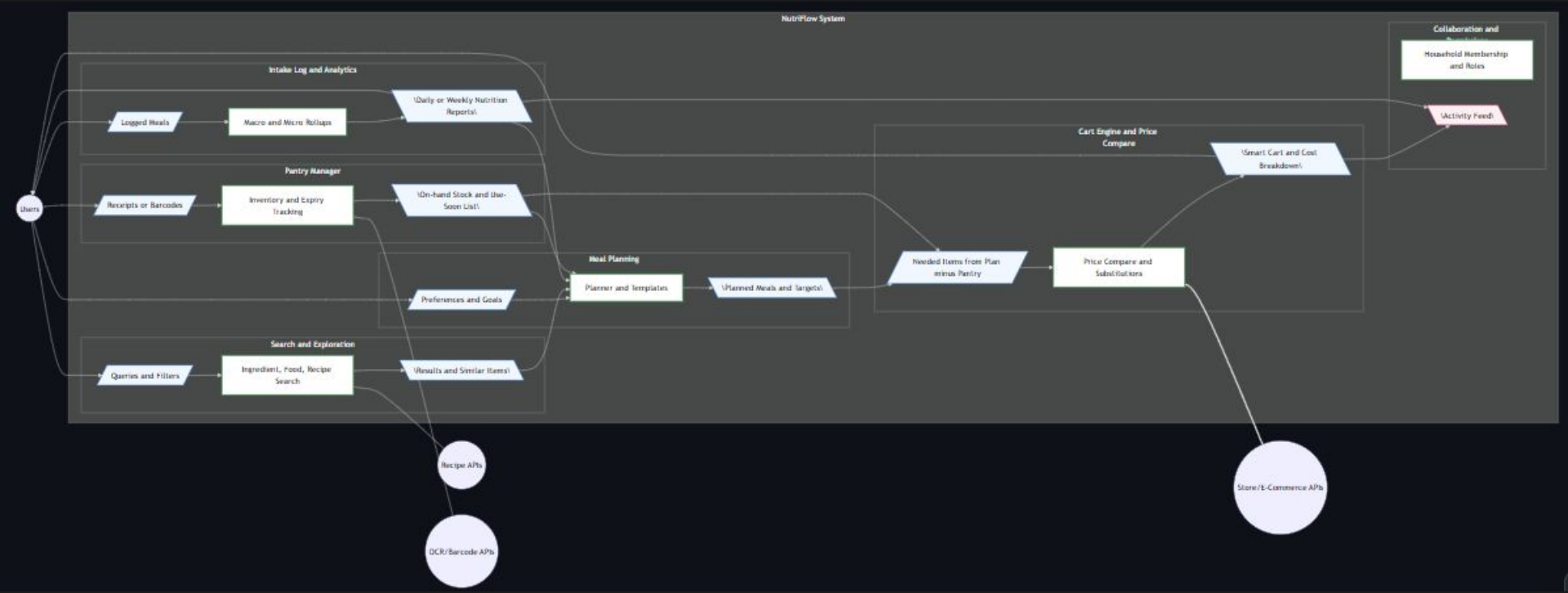
Design Specifications

Design Diagrams - Conceptual/Data Model



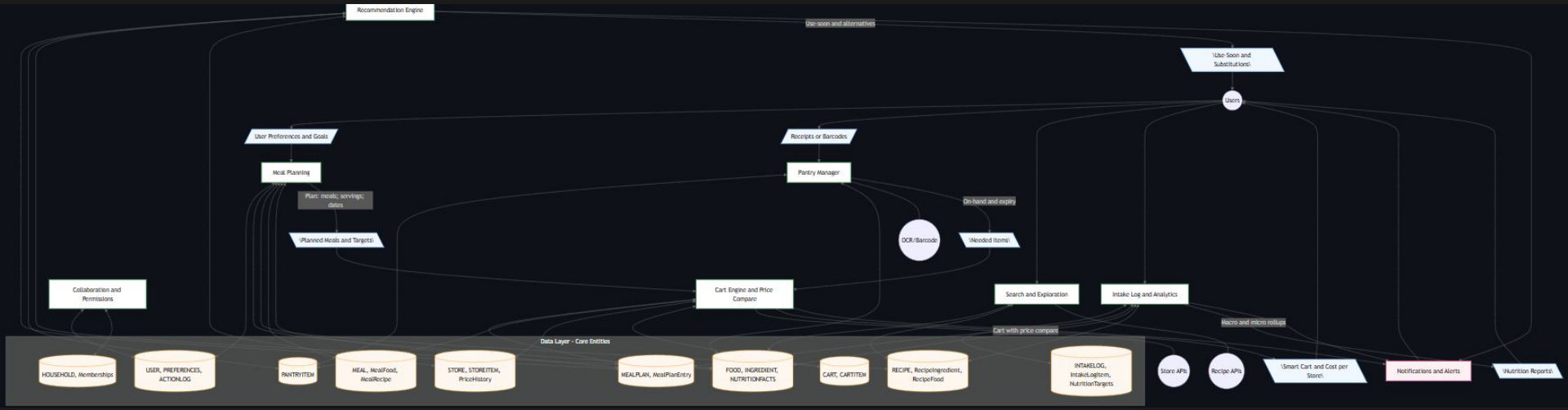
Interactive Diagram Link: https://github.com/randalhucker/nutriflow/blob/main/cs5001/assignment_4/Design_Diagram.md

Design Diagrams - Conceptual/Data Model



Interactive Diagram Link: https://github.com/randalhucker/nutriflow/blob/main/cs5001/assignment_4/Design_Diagram.md

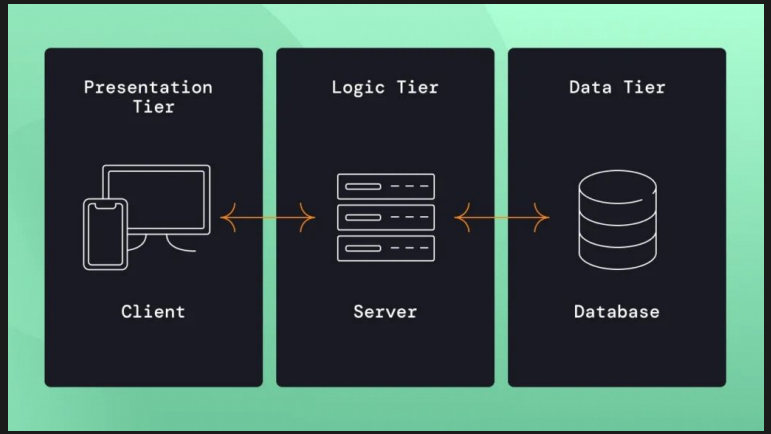
Design Diagrams - Conceptual/Data Model



Interactive Diagram Link: https://github.com/randalhucker/nutriflow/blob/main/cs5001/assignment_4/Design_Diagram.md

System Architecture

- NutriFlow uses a **multi-client, three-tier architecture** consisting of:
  - A **web-based Single Page Application (SPA)** for desktop/laptop users
  - A **native mobile client (React Native / Expo)** for iOS and Android
  - A centralized **backend API server** responsible for authentication, business logic, API, and database interaction
  - A dedicated **database layer** for persistent storage
- Both frontend clients communicate exclusively through the same REST API, enabling shared data models and consistent behavior across platforms
- This design allows NutriFlow to support multiple user interfaces while maintaining a single source of truth for application logic and data.



Client (Presentation) Layer, Web + Mobile

- NutriFlow supports two frontend implementations:
  - **Web SPA**
    - Runs in the browser
    - Handles client-side routing and UI state
    - Optimized for desktop workflows (meal planning, bulk editing)
  - **Mobile Native Client**
    - Runs on mobile devices
    - Uses native platform UI components
    - Optimized for on-the-go usage (daily intake logging, quick edits
- Both clients implement:
  - CRUD workflows for foods, meals, recipes, and meal plans
  - Local UI state management
  - API-driven data fetching and caching
- This ensures feature parity and minimizes duplicated logic.

Backend/Data Layer

Backend Layer

- **Web Asset Host**
  - Serves the compiled web frontend bundle
  - Enables a unified deployment pipeline’
- **API Provider**
  - Exposes REST endpoints for all domain entities (foods, meals, meal plans, pantry, intake logs)
- **Business Logic**
  - Load balancing
  - Building Store/Ingredient/Recipe databases based on user requests
- This backend acts as the application layer, coordinating all interactions between clients and the database.

Data Layer

- Persistent storage for:
  - User accounts
  - Foods and nutrition metadata
  - Meals and recipes
  - Meal plans
  - Pantry contents
  - Daily intake logs
- The schema is designed around relational consistency between entities, allowing NutriFlow to model the full food lifecycle:
  - **Planning → Preparation → Consumption → Procurement**
- This structure enables higher-level features such as pantry-aware cart generation and macro-based planning.

Technologies

Technologies

- UI
  - **Typescript** frontend to support robust typing and chosen frontend library
  - **React Native** to support simultaneous web and mobile platform development
  - **Nativewind/Gluestack** to allow us to create reusable custom-styled components
  - **TS-Rest/react-query** to allow seamless API integration with React's state management paradigm
- Backend/Database
  - **Typescript** backend for type consistency across layers of the application
  - **OpenAPI** specification supporting swaggerUI pages for API documentation/testing
    - We also use **Postman** for API testing
  - **Zod and TS-Rest/core** to support the creation of a strongly-typed API contract
    - We developed a **custom npm package** used to distribute said contract to the UI
  - **PostgreSQL** for relational database management
  - **Docker** to containerize deployments and quickly spin up development builds

Milestones

Milestones

Completed

- M1 - Project Kickoff & Architecture Baseline (09-15-2025)
- M2 - Core Data Layer & Auth Complete (10-20-2025)
- M3 - External Integrations Alpha (12-14-2025)

In Progress

- M4 - Cart Engine + Smart Cart UI Beta (02-24-2025)
- M5 - Search, Planning, and Intake UI Cohesion (03-16-2026)

Final Deliverables

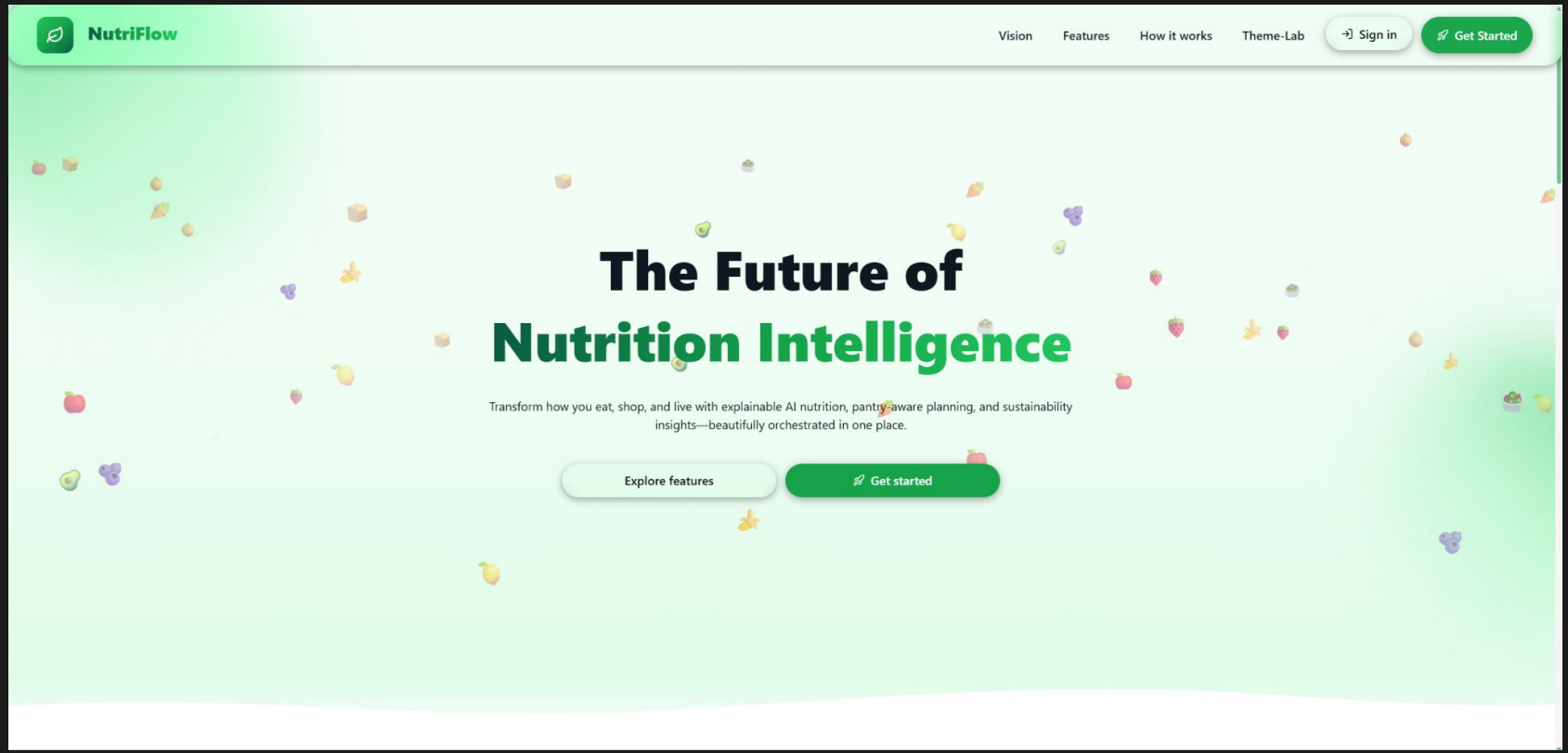
- M6 - QA Coverage, Security & Docs (03-23-2026)
- M7 - MVP Release Candidate & Submission (03-30-2026)
- M8 - AI/ML Touchpoints POC? (04-07-2026)

Results

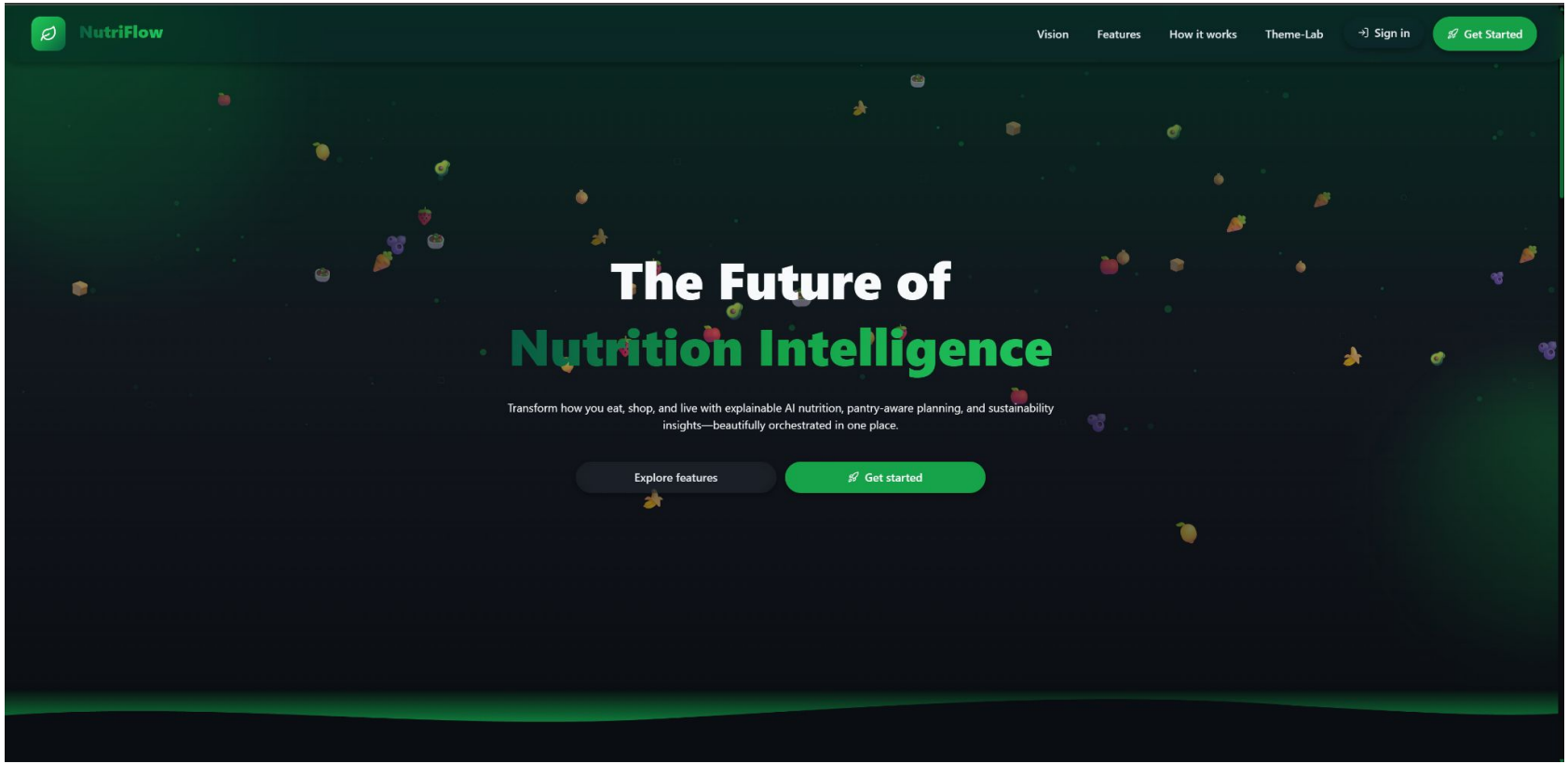
Current Progress

- Solid system architecture and data model design
- Fully functional backend API supporting CRUD for core models as well as additional logic
- Unified Mobile and Web UI is functional but still under heavy development
  - Dark mode support has been added to avoid heavy refactor in the future
- Auth flow and basic page routing is completed

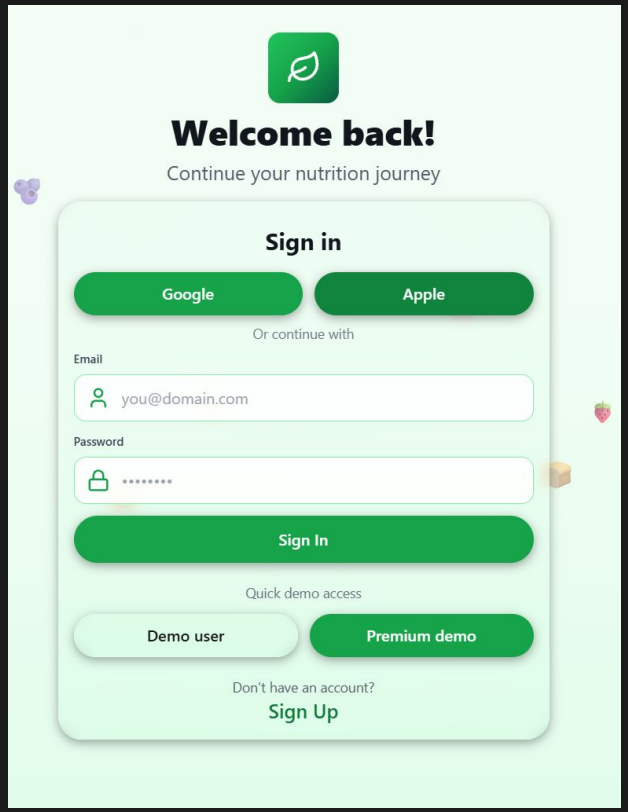
Implemented Page Examples

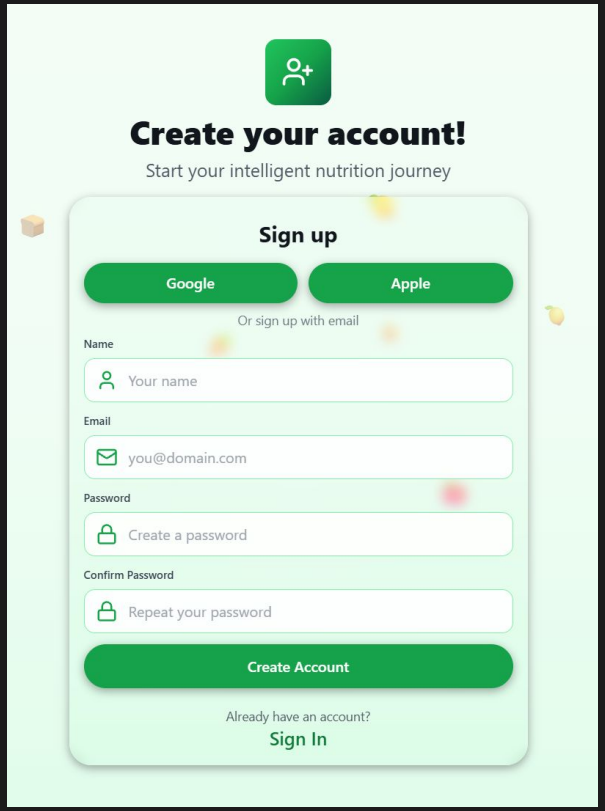


Implemented Page Examples

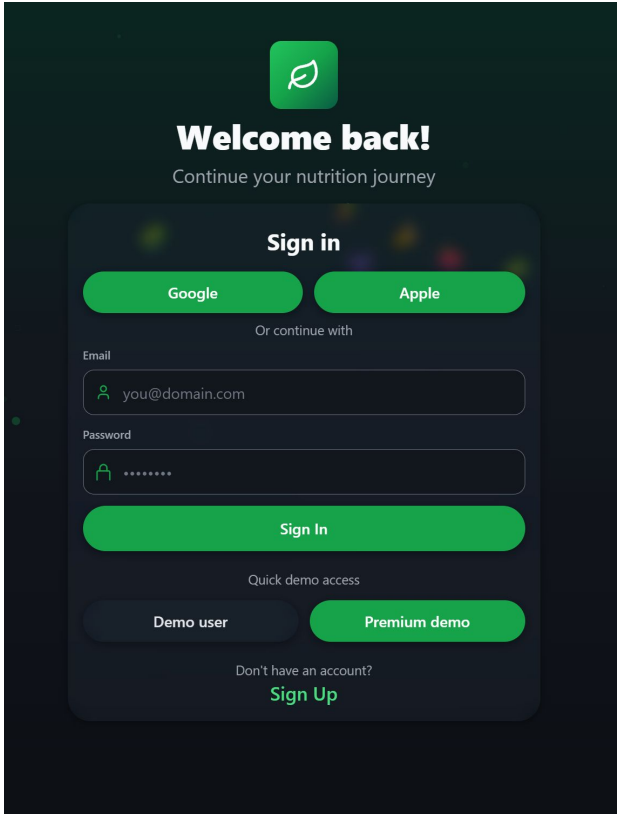


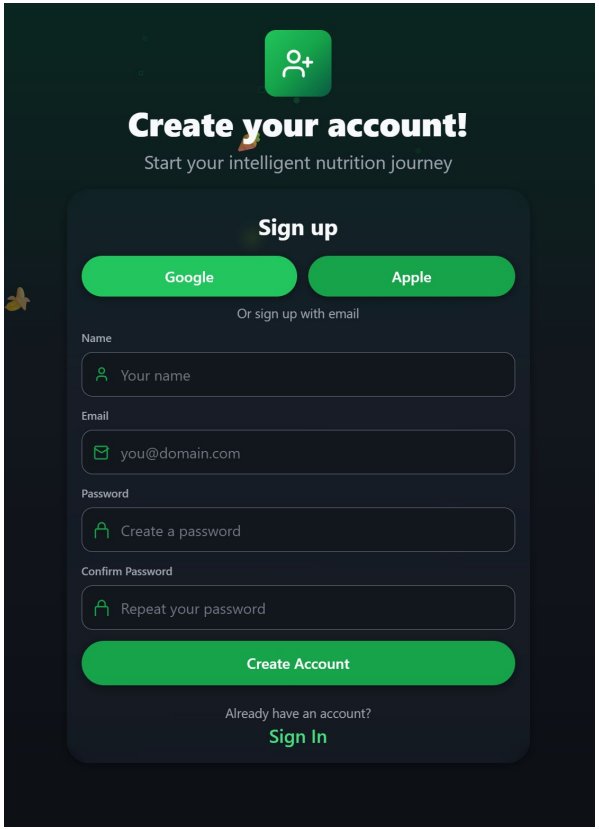
Implemented Page Examples





Implemented Page Examples





Next Steps

- Continue filling out the application's UI
  - Established a good style and set of base components
  - All that's left is to implement the pages and hook them up to the API
- Develop Pantry/Cart Management Algorithm Progress
  - Calculate groceries needed for meal plan factoring in current stock
  - Develop recommendation algorithms (cheapest/healthiest/combination)
- Iterate on the UX to improve and streamline the application
  - Initial mocks serve as a general guide, but once features are fully implemented it may need to be tweaked
- Testing and User Documentation
  - Write appropriate tests and complete documentation so Nutriflow is ready for use come April
- Prepare Demo flow and test data
  - We want to efficiently demonstrate as much functionality as possible

Challenges

Challenges

Ensuring consistent behavior across web SPA and native mobile environments

Solution:
Platform-specific component variants when needed, shared API contract, and pulling central business logic into the backend

Maintaining smooth UI updates while interacting with remote APIs

Solution:
Implement client-side caching and modularize frontend state management

Balancing ambitious feature goals with realistic capstone timeline

Solution:
Focus on building a strong and extensible architectural/data model foundation, prioritize MVP functionality and defer advanced features to post-graduation timeline

Thank you!